4.6 40-Entry Unified Out-of-Order Scheduler and Integer Execution Unit for the AMD Bulldozer x86-64 Core

Michael Golden, Srikanth Arekapudi, James Vinh

AMD, Sunnyvale, CA

AMD's two-core Bulldozer module [1,2] implements the AMD x86-64 microarchitecture in an 11-layer 32-nm SOI HKMG technology. The 40-instruction out-of-order unified integer scheduler issues up to four operations per cycle and supports single-cycle wake-up of dependent operations. The 2.37mm^2 integer execution unit supports single-cycle data bypass among four independent functional units. Compared to previous AMD x86-64 cores [3-6], project goals reduce the number of FO4 inverter delays per cycle by more than 20%, while maintaining constant IPC, to achieve higher frequency and performance in the same power envelope, even with increased core counts.

Critical paths (Fig 4.6.1) are implemented without exotic circuit techniques or heavy reliance on full-custom design. Dynamic logic appears only when required for density. Inputs and outputs of dynamic gates tolerate duty-cycle variation by flowing through timing test points if clock arrives early. In lieu of dynamic logic, extensive use of skewed static CMOS standard-cell gates speeds transmission of the evaluate edge versus the pre-charge edge. To minimize power consumption, the architecture favors tag movement relative to data movement and pointer management relative to shifting, collapsing data structures. Duplicated register files and functional units ameliorate wire-limited critical paths.

The integer scheduler issues up to four out-of-order operations per cycle. Unlike a reservation station, which stores data or tags, the scheduler stores only tags: the physical register numbers (PRN) of destination and source operands. To save power, all structures except the physical register file (PRF) and functional units manipulate 8-bit PRNs instead of 64-bit instruction data. Instead of a power-hungry shifting, collapsing structure to preserve age information, an ancestry table keeps track of the oldest instruction in the scheduler. The oldest instruction will be picked, if ready. If not, a priority encoder scans ready instructions in their physical order, which does not correspond to their age order.

When an operation is picked for execution, the destination PRN is read from a register file and broadcast to a fully associative CAM, which stores up to four source operands for every instruction in the scheduler (Fig. 4.6.2). To save power, pairs of destination bits are decoded before broadcast, reducing the number of switching bitlines. An extra pull-down drives PRNs of results from the load/store unit into the source CAM. Rather than pulling down on a "mismatch" signal if a single bit mismatches in the CAM, a logical AND structure is used. This has two benefits. Fewer CAM structures switch – match is less likely than mismatch – and, because it is known *a priori* whether a source will match a destination from an arithmetic logic unit (ALU) or an address generation unit (AGU), a single CAM wire can share pulldowns between AGU and ALU destinations.
One hundred sixty "current match" signals indicating which source operands match the destinations picked in the current cycle flow through glitch latches into post-wake and pick logic (Fig. 4.6.3). Post-wake and pick logic are implemented with static CMOS logic, with beta ratios skewed higher or lower than the natural process beta ratio (Fig. 4.6.4). Skewed gates favor the evaluate edge of the glitch latch output over the reset edge. Although wide OR functions in the post-wake and pick logic are amenable to domino logic, re-clocking the glitch latch outputs would make the design susceptible to duty-cycle variation. Source readiness information flowing in from register rename logic depends on pick results from the previous cycle, and is also critical. Though standard cell flops launch this path, the data is gated with the clock and fed into a glitch latch local to the picker to allow it to flow quickly through the skewed gates.

The pick result must be clocked and sent back to the wake-up array for the next cycle. A low-latency edge-triggered clock gater reduces clocking overhead (Fig. 4.6.5). The device has short setup time, determined only by having the output of the data inverter switch past the input. The cross-coupled PMOS keeper devices allow the dynamic node to recover from any false evaluation before the input switches full-rail, and do not fight the NMOS evaluation devices. A self-resetting NMOS keeper holds data for a full clock phase. The fast combination of a two-high NMOS stack followed by an inverter determines the clock-to-output latency.

The pick signal is driven directly into the instruction payload register file (PLDRF) as a read word line. Operand source PRNs in PLDRF flow out of glitch latches into dynamic bypass compare logic. Although most bit reads in the payload are single-ended, source PRNs are read dual-rail and full-swing. This (1) allows the use of cross-coupled keepers, against which evaluate pull-downs do not have to fight; (2) eliminates inverters in data paths; (3) permits the use of skewed logic gates for both true and complementary data; and, (4) increases tolerance of duty-cycle variation by allowing late-arriving bits to flow through clock gaters.

The execution unit supports single-cycle operand bypass from an instruction to a dependent instruction. Two ALU ops and two AGU ops can be executed in a cycle. AGU ops include increment/decrement (INC), address generate, and x86-64 LEA instructions. The ALUs and the INC units participate in single-cycle operand bypass. Four result buses, one from each of the units, feed four writes ports into the PRF. Each execution unit can receive data from any of the four write ports or from a PRF read. Each execution unit has two sources, requiring eight read ports in the PRF. To remove wire delay (and congestion) from the critical bypass path, the PRF is duplicated. The register file is divided into halves; the critical half, further from the output driver, has fewer pull-downs on the local bitline, reducing slew rates on critical words with minimal area impact (Fig. 4.6.6). Duplicating the INC removes the vertical wire delay from the worst-case bypass path, which is from any INC to the far ALU.

The ALU itself consists of multiple blocks, each of which executes a family of ALU ops. The results of these blocks are multiplexed hierarchically onto the ALU result bus, implemented in footless domino logic. Results from the adder and shifter are timing-critical and pull down directly on the results bus [1]. Like data results, flag results also have a single-cycle loop. Flag logic generates carry, overflow, sign, and auxiliary flags in parallel to the data result, and zero and parity flags from the final data result. Flag bypass is simpler because only ALUs produce and consume flags.

The shift unit supports all shift and rotate operations with a footless domino barrel-shifting core that performs a rotate operation through the carry bit. Set-up of data to the clock gaters driving into the shifter, which launch data on the falling edge of the clock, is susceptible to duty-cycle variation. To ameliorate this problem, the critical six bits that drive into the shift count decoder that controls the barrel shifter are driven dual-rail throughout the entire EX datapath, including the PRF read. All other functional units in the AGU and the ALU are built from standard cell logic. This includes the way predictor array (Fig. 4.6.4). The critical INC and adder units drive data onto the results bus using the same edge-triggered clock gater used in the picker.

Acknowledgements:
This design is the result of years of effort by AMD's entire Bulldozer team.

References:
[1] Fischer, T.; Arekapudi, S.; Dietz, C.; Golden, M.; Hilker, S.; Hurd, K.; Johnson, D.; McIntyre, H.; Vinh, J.; White, J.; Wilcox, K.; "Design Solutions for the Bulldozer 32nm SOI 2-core processor module in an 8-core Orochi CPU," *submitted to ISSCC 2011.*
[2] Butler, M.; "AMD 'Bulldozer' Core - a new approach to multithreaded compute performance for maximum efficiency and throughput," Hot Chips 22, Aug. 2010.
[3] Jotwani, R.; Sundaram, S.; Kosonocky, S.; Schaefer, A.; Andrade, V.; Constant, G.; Novak, A.; Naffziger, S.; "An x86-64 core implemented in 32nm SOI CMOS," *ISSCC Dig. Tech. Papers*, pp. 106-107, Feb. 2010.
[4] Dorsey, J.; Searles, S.; Ciraula, M.; Johnson, S.; Bujanos, N.; Wu, D.; Braganza, M.; Meyers, S.; Fang, E.; Kumar, R.; , "An Integrated Quad-Core Opteron Processor," *ISSCC Dig. Tech. Papers*, pp. 102-103, Feb. 2007.
[5] Golden, M.; Arekapudi, S.; Dabney, G.; Haertel, M.; Hale, S.; Herlinger, L.; Kim, Y.; McGrath, K.; Palisetti, V.; Singh, M.; "A 2.6GHz Dual-Core 64bx86 Microprocessor with DDR2 Memory Support," *ISSCC Dig. Tech. Papers*, pp. 325-332, Feb. 2006.
[6] Keltcher, C.N.; McGrath, K.J.; Ahmed, A.; Conway, P.; , "The AMD Opteron processor for multiprocessor servers," *Micro, IEEE* , vol. 23, no. 2, pp. 66- 76, March-April 2003.

 978-1-61284-302-5/11/$26.00

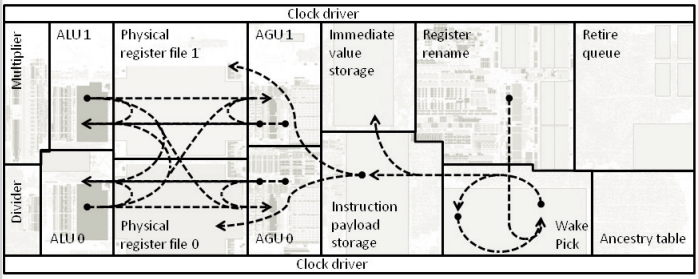

Figure 4.6.1: Floorplan plot showing critical paths.

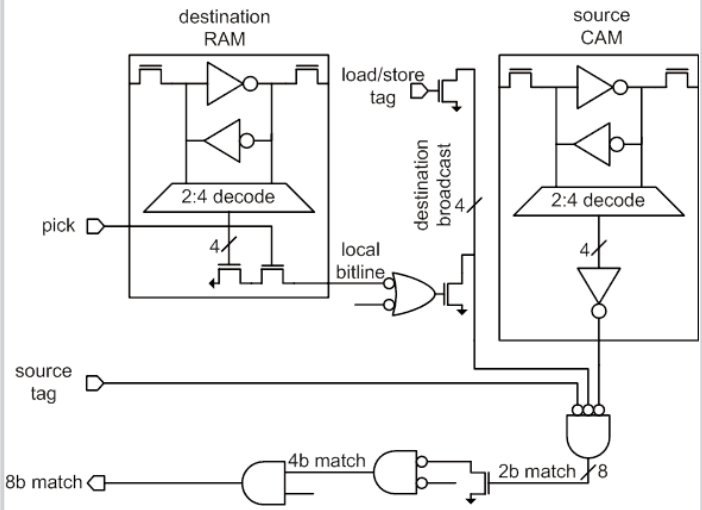

Figure 4.6.2: Wake-up array logic.

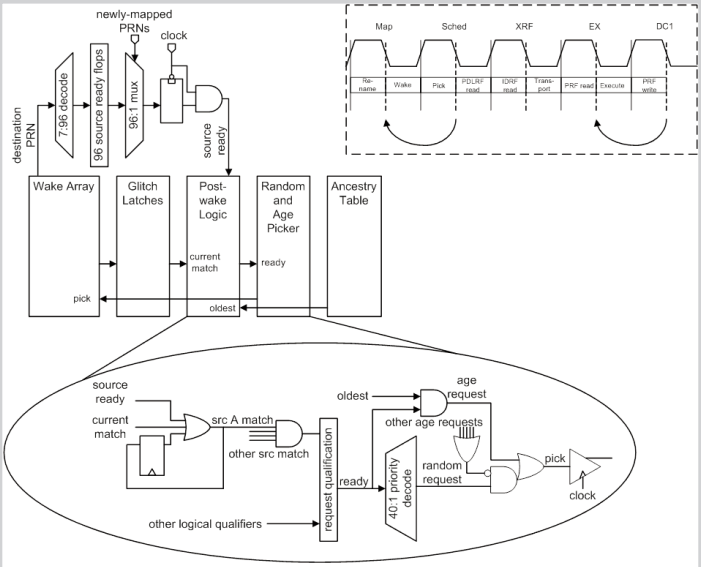

Figure 4.6.3: Picker logic.

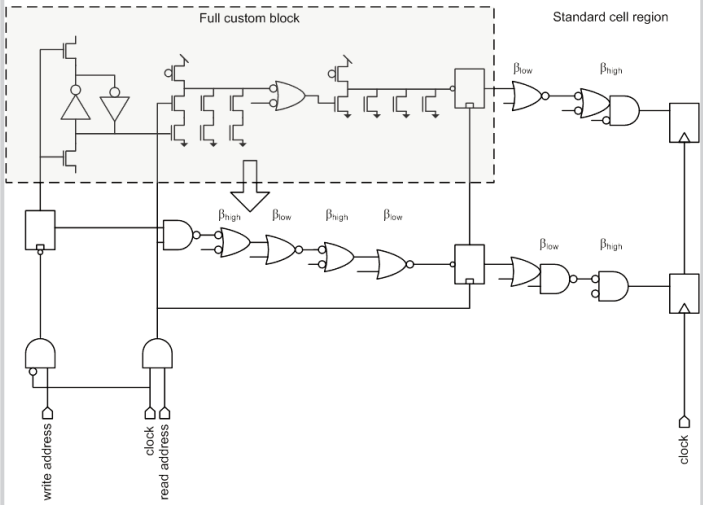

Figure 4.6.4: Array implemented as standard cells.

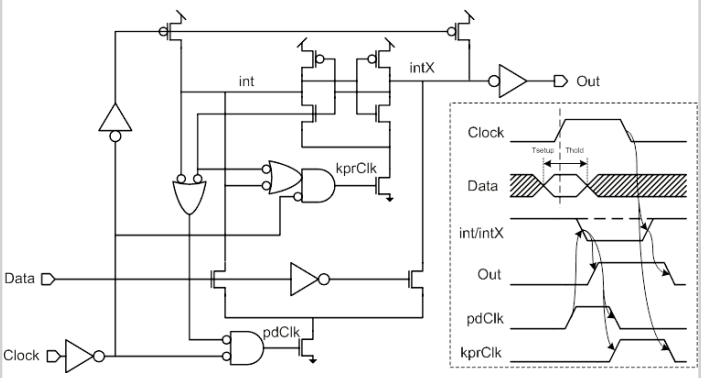

Figure 4.6.5: Edge-triggered low-latency clock gater.

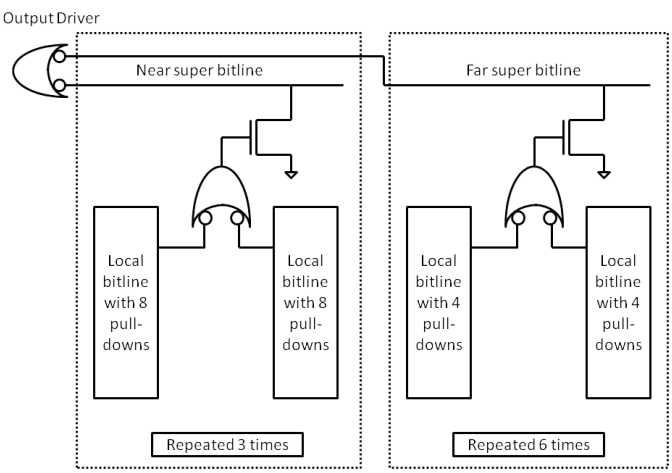

Figure 4.6.6: Asymmetric local bitlines in physical register file.

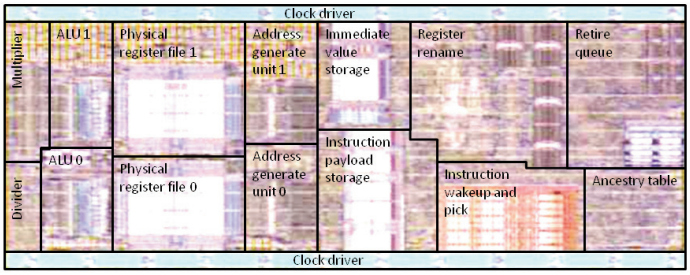

Figure 4.6.7: Die photograph showing floorplan.

978-1-61284-302-5/11/$26.00